Notification of Revoked Access
Tokens in the ACE Framework

draft-tiloca-ace-revoked-tokens-notification-06

Marco Tiloca, RISE
Ludwig Seitz, Combitech
Francesca Palombini, Ericsson
Sebastian Echeverria, CMU SEI
Grace Lewis, CMU SEI

IETF 112, ACE WG, November 9th, 2021
Recap

› An Access Token may be revoked, before expiration
  – Client/RS has been compromised, or decommissioned
  – Changed access policies or outcome of their evaluation
  – Changed ACE profile to use

› Token introspection at the AS is available only for the RS
  – Validate one Access Token at the time

› Contribution: new interface at the Authorization Server (AS)
  – The AS maintains one Token Revocation List (TRL) resource
  – The TRL contains the hashes of revoked, not-yet-expired tokens
  – C/RS can GET or GET-Observable from the TRL
  – C/RS retrieve only their own pertaining portion of the TRL

› Benefits
  – Complement token introspection
  – No need for new endpoints at C or RS
How it works

› Token hashes computed as per RFC 6920 (binary format)
  – Hash input: what is in ‘access_token’ of the AS response from /token

› TRL resource at the AS
  – CBOR array of Token hashes
  – Add token hashes when Tokens are revoked
  – Remove token hashes when revoked Tokens expire

› Interaction
  – C and RS get the URL to the TRL endpoint upon registration
  – C and RS obtain only hashes of their own pertaining Tokens
  – A registered Administrator gets all Token hashes in the TRL
Modes of operation

› **Common features**
  – Response limited to the portion of the TRL pertaining the requester
  – TRL filtering based on authenticated identity of the requester (secure session)

› **Full Query - GET [Observe: 0] coaps://example.as.com/revoke/trl**
  – Get all the pertaining token hashes in the TRL
  – The AS MUST support it

› **Diff Query - GET [Observe: 0] coaps://example.as.com/revoke/trl?diff=3**
  – Get the N most recent, pertaining updates to the TRL
  – The AS MAY support it

› **STP-based query** – Appendix B
  – Extends the two modes above, using the Series Transfer Pattern (STP)
  – Enables trasferring of TRL updates in chunks, from a “resumption point”
  – Based on a review from Carsten Bormann and on input from Ben Kaduk
Updates from -04, -05 and -06

› Early clarifications, at protocol overview
  – What the different modes of operations offer
  – The registration process at the AS is out of scope in ACE

› Added error handling at the AS

› Optional “pmax” attribute when observing, see draft-ietf-core-conditional-attributes
  – No more than pmax seconds between two consecutive observe notifications

› Response format and processing for the STP-based query mode
  – New content format application/ace-trl+cbor and new registry “Token Revocation List”
  – Response payload as a CBOR map

› Addressed comments on -04 from Michael Richardson [1] – Thanks!
  – Observation as subscription; difference from per-Token introspection; requirements for C/RS

› Editorial improvements in IANA considerations

[1] https://mailarchive.ietf.org/arch/msg/ace/TYfW7aT8dR7sXDvlcJfHOVTJWeA/
Summary and next steps

› Notification of revoked Access Token
  – GET or GET-Observable; for both Client and Resource Server
  – (i) Full query; (ii) Diff query; (iii) Query with Series Transfer Pattern (STP)

› Version -06 is stable and incorporates:
  – Error handling and response payload in the STP-based query mode
  – Comments from Michael Richardson on -04
  – Review from Carsten Bormann and comments from Ben Kaduk on -01
  – Earlier review from Travis Spencer and comments from Jim Schaad

› Next steps
  – STP-based query mode in the document body

› WG adoption?
Thank you!

Comments/questions?

https://gitlab.com/crimson84/draft-tiloca-ace-revoked-token-notification
Backup
Protocol overview

```

Authorization Server

revoke/trl  TRL: {th1,th2,th3}

th1,th2,th3  th1,th2  th1  th3  th2,th3
v  v  v  v  v

Administrator  Client 1  Resource 1  Client 2  Resource 2

:  :  t1  :  :  :  :  t3  :
:  ::  ::  ::  ::  ::  ::
:  ::  ::  :
:  ::  ::  :
:  ::  ::  :
:  ::  ::  :
:  ::  ::  :
```


Example with Full Query

RS

Registration: POST

AS

<

2.01 CREATED
Payload: {
  ...
  "trl" = "revoke/trl",
  "trl_hash" = "sha-256",
  "n_max" = 10
}

GET Observe: 0
coap://example.as.com/revoke/trl/

<

2.05 CONTENT Observe: 42
Payload: []
.
.
.

(Access Tokens t1 and t2 issued and successfully submitted to RS)
.
.

IETF 112 | ACE WG | 2021-11-09 | Page 10
Example with Full Query (ctd.)

(Access Token t1 is revoked)

2.05 CONTENT Observe: 53
Payload: [bstr.h(t1)]
  ...

(Access Token t2 is revoked)

2.05 CONTENT Observe: 64
Payload: [bstr.h(t1),
         bstr.h(t2)]
  ...

(Access Token t1 expires)

2.05 CONTENT Observe: 75
Payload: [bstr.h(t2)]
  ...

(Access Token t2 expires)

2.05 CONTENT Observe: 86
Payload: []
Types of TRL queries

› **Common features**
  – Limited to the portion of the TRL pertaining the requester
  – TRL filtering based on authenticated identity of the requester (secure session)

› **Full Query** – `GET [Observe: 0] coaps://example.as.com/revoke/trl`
  – Request for all pertaining token hashes in the TRL
  – Return a CBOR array, with the Token hashes as elements

› **Diff Query** – `GET [Observe: 0] coaps://example.as.com/revoke/trl?diff=3`
  – Request for the latest N updates to the pertaining portion of the TRL list
  – Build N entries as CBOR arrays. Each entry refers to an update and has:
    › An element “deleted”, with a CBOR array of Token hashes.
    › An element “added”, with a CBOR array of Token hashes.
  – Return a CBOR array with the N arrays as element, in reverse chronological order

› **STB-based Query – Appendix B**
  – Builds on and extends the Full Query and Diff Query modes
  – Uses the Series Transfer Pattern (STB), to enable transfers in chunks from a “resumption point”
STP-based query mode

- Rather than the N most recent TRL updates...
  - Get N updates from “where we stopped last time”
  - Revert to Full Query if not possible, e.g., information loss/removal at the AS

- Use the Series Transfer Pattern (STP) and its “Cursor” pattern
  - Both (a) Full Query and (b) Diff Query requests return also a cursor
  - (a) Pointer to the most recent, pertaining TRL update
  - (b) Pointer to the most recent TRL update included in the response

- In this “enhanced Diff Query” mode
  - A follow-up request may resume from after the cursor
  - Adjacent batches of TRL updates are possible, limiting excessive latencies

- Handled corner cases
  - No updates, or no updates after the cursor
  - Requested updates have been deleted as too old